A. Objective Gradients

To compute the gradient $\nabla\Phi(\theta)$, we first write the gradient in terms of the gradients of $Z_t(\theta)$ and π_θ:

$$\nabla\Phi(\theta) = \sum_{t=1}^{T}\left[\frac{1}{Z_t(\theta)}\sum_{i=1}^{m}\frac{\nabla\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}r(\mathbf{x}_t^i,\mathbf{u}_t^i) - \frac{\nabla Z_t(\theta)}{Z_t(\theta)^2}\sum_{i=1}^{m}\frac{\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}r(\mathbf{x}_t^i,\mathbf{u}_t^i) + w_r\frac{\nabla Z_t(\theta)}{Z_t(\theta)}\right].$$

From the definition of $Z_t(\theta)$, we have that

$$\frac{\nabla Z_t(\theta)}{Z_t(\theta)} = \frac{1}{Z_t(\theta)}\sum_{i=1}^{m}\frac{\nabla\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}.$$

Letting $\tilde{J}_t(\theta) = \frac{1}{Z_t(\theta)}\sum_{i=1}^{m}\frac{\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}r(\mathbf{x}_t^i,\mathbf{u}_t^i)$, we can rewrite the gradient as

$$\begin{aligned}\nabla\Phi(\theta) &= \sum_{t=1}^{T}\frac{1}{Z_t(\theta)}\sum_{i=1}^{m}\frac{\nabla\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}\left[r(\mathbf{x}_t^i,\mathbf{u}_t^i) - \tilde{J}_t(\theta) + w_r\right] \\ &= \sum_{t=1}^{T}\frac{1}{Z_t(\theta)}\sum_{i=1}^{m}\frac{\pi_\theta(\zeta_{i,1:t})}{q(\zeta_{i,1:t})}\nabla\log\pi_\theta(\zeta_{i,1:t})\xi_t^i,\end{aligned}$$

using the identity $\nabla\pi_\theta(\zeta) = \pi_\theta(\zeta)\nabla\log\pi_\theta(\zeta)$. When the policy is represented by a large neural network, it is convenient to write the gradient as a sum where the output at each state appears only once, to produce a set of errors that can be fed into a standard back-propagation algorithm. For a neural network policy with uniform output noise σ and mean $\mu(\mathbf{x}_t)$, we have

$$\begin{aligned}\nabla\log\pi_\theta(\zeta_{i,1:t}) &= \sum_t \nabla\log\pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \\ &= \sum_t \nabla\mu(\mathbf{x}_t)\frac{\mathbf{u}_t - \mu(\mathbf{x}_t)}{\sigma^2},\end{aligned}$$

and the gradient of the objective is given by

$$\nabla\Phi(\theta) = \sum_{t=1}^{T}\sum_{i=1}^{m}\nabla\mu(\mathbf{x}_t^i)\frac{\mathbf{u}_t^i - \mu(\mathbf{x}_t^i)}{\sigma^2}\sum_{t'=t}^{T}\frac{1}{Z_{t'}(\theta)}\frac{\pi_\theta(\zeta_{i,1:t'})}{q(\zeta_{i,1:t'})}\xi_{t'}^i.$$

The gradient can now be computed efficiently by feeding the terms after $\nabla\mu(\mathbf{x}_t^i)$ into the standard back-propagation algorithm.

B. Dynamic System Descriptions

This appendix describes the dynamical systems corresponding to the simulated robots in the swimming, hopping, and walking tasks. Images of each robot are provided in Figure 1 of the paper.

Swimmer: The swimmer is a 3-link snake, with 10 state dimensions for the position and angle of the head, the joint angles, and the corresponding velocities, as well as 2 action dimensions for the torques. The surrounding fluid applies a drag on each link, allowing the snake to propel itself. The simulation step is 0.05s, the reward weights are $w_\mathbf{u} = 0.0001$, $w_v = 1$, and $w_h = 0$, and the desired velocity is $v_x^\star = 2$m/s.

Hopper: The hopper has 4 links: torso, upper leg, lower leg, and foot. The state has 12 dimensions, and the actions have 3. To make it easier to optimize a gait with DDP, we employed a softened contact model as proposed in (Tassa et al., 2012). The reward weights are $w_\mathbf{u} = 0.001$, $w_v = 1$, and $w_h = 10$, and the desired velocity and height are $v_x^\star = 1.5$m/s and $p_y^\star = 1.5$m. A lower time step of 0.02s was used to handle contacts.

Walker: The walker has 7 links, corresponding to two legs and a torso, 18 state dimensions and 6 torques. The reward weights are $w_\mathbf{u} = 0.0001$, $w_v = 1$, and $w_h = 10$, and the desired velocity and height are $v_x^\star = 1.2$m/s and $p_y^\star = 1.5$m. The time step is 0.01s.

3D Humanoid: The humanoid consists of 13 links, with a free-floating 6 DoF base, 4 ball joints, 3 joints with 2 DoF, and 5 hinge joints, for a total of 29 degrees of freedom. Ball joints are represented by quaternions, while their velocities are represented by 3D vectors, so the entire model has 63 dimensions. The reward weights are are $w_\mathbf{u} = 0.00001$, $w_v = 1$, and $w_h = 10$, and the desired velocity and height are $v_x^\star = 2.5$m/s and $p_y^\star = 0.9$m. The time step is 0.01s. Due to the complexity of this model, the joint noise was reduced from 10% of example torque variance to 1%.